



ÉCOLE SUPÉRIEURE D'INFORMATIQUE

# Project Java /POO/UML

---

LORANN



AL KHOURY ALEXI  
AMBROSINI MELINE  
FLEURY PIERRE  
PARVILLERS GAUTHIER

EXIA.CESI | ROUEN

## Table of contents

Introduction .....	2
Context .....	2
Requests .....	2
Constraints .....	2
Roles .....	2
Provisional schedule .....	2
Solution .....	3
Modelling .....	3
Package diagram .....	3
Class diagram .....	3
Contract class diagram .....	4
Controller class diagram .....	4
View class diagram .....	5
Model class diagram .....	5
Main class diagram .....	6
Components diagram .....	6
Sequence diagram .....	6
Code .....	6
Documentation .....	9
Javadoc .....	9
JXR report .....	9
SureFire report .....	9
Results .....	9
Effective schedule .....	9
Team review .....	10
Personal review .....	10
Méline Ambrosini .....	10
Alexi Al Khoury .....	10
Pierre Fleury .....	10
Gauthier Parvillers .....	10
Ways of improvement .....	10

# Introduction

## Context

The project's goal is to rewrite an abandonware : LORANN. It's a PacMan-style game in which the hero, Lorann, evolves in the Nova-Ann's world. His goal is to free 101 crypts fell under the domination of Nekron's mask. However, crypts are protected by 4 monsters, sent by Nekron which try to catch Lorann. Luckily, Lorann can destroy his ennemies with a special spell. During his quest, Lorann has to pick up as many treasures as possible and has to reach an energy bubble which allows to chase Nekron's mask from the crypt and open a gate that will lead Lorann to the next level.

## Requests

We have to replicate this game in 5 levels. We must be able to reach these levels via a parameter in the source code, a configuration file, or a record in a database... Moreover, levels have to be stored in a database. We also have to implement the emblematic graphic elements of the original game. Besides, we must do test units and provide all the documentation related to the code.

## Constraints

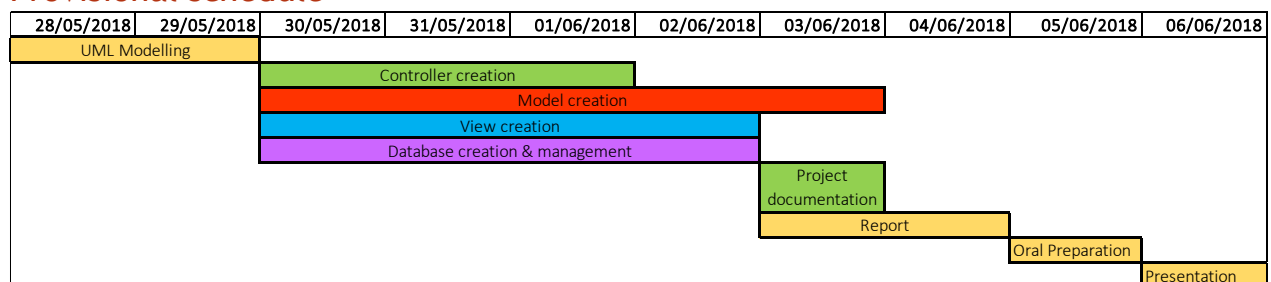
Some constraints have to be respected :






- The use of Java, Maven, Git and JUnit is mandatory
- Use only Swing to provide a graphical user interface
- No SQL queries in the Java source code

## Roles

<b>Méline AMBROSINI</b>	<b>Project manager, in charge of the view package</b>
<b>Alexi AL KHOURI</b>	In charge of the database creation and the levels' storage
<b>Pierre FLEURY</b>	In charge of the contract package
<b>Gauthier PARVILLERS</b>	In charge of the model package

## Provisional schedule



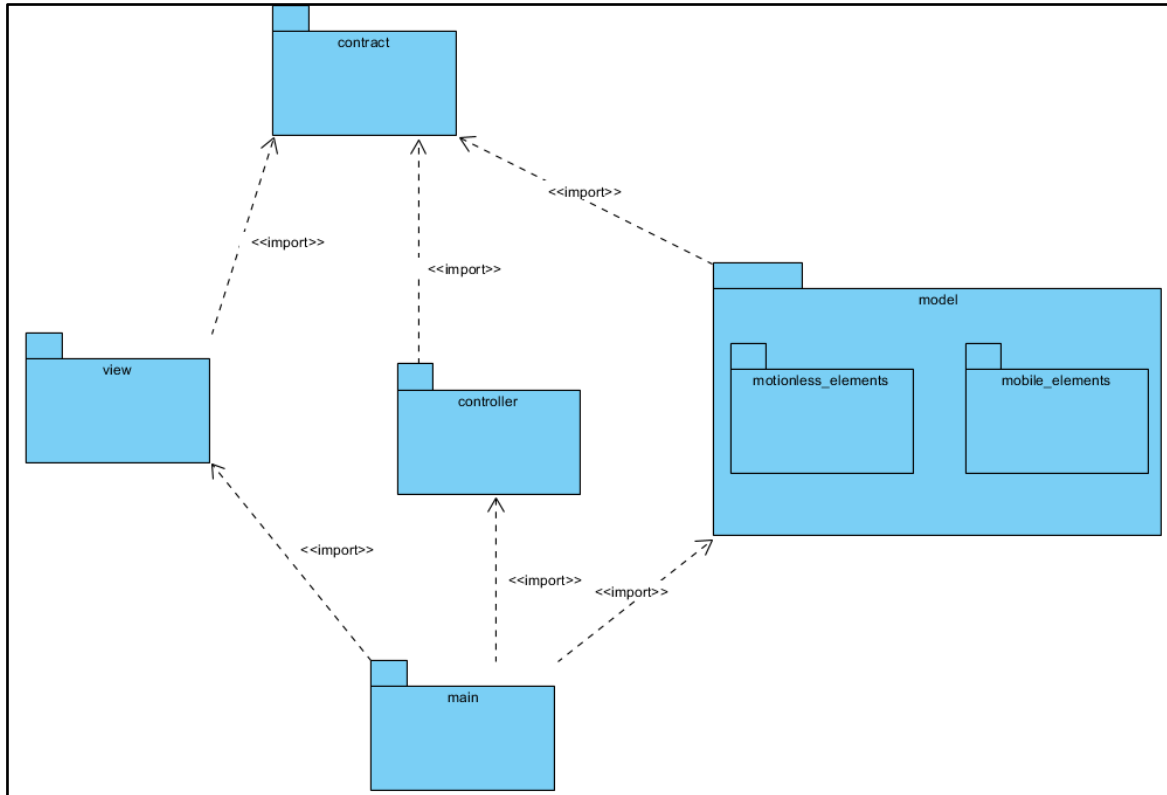
Roles	
	Team
	Alexi AL KHOURY
	Méline AMBROSINI
	Pierre FLEURY
	Gauthier PARVILLERS

## Solution

### Modelling

#### Package diagram

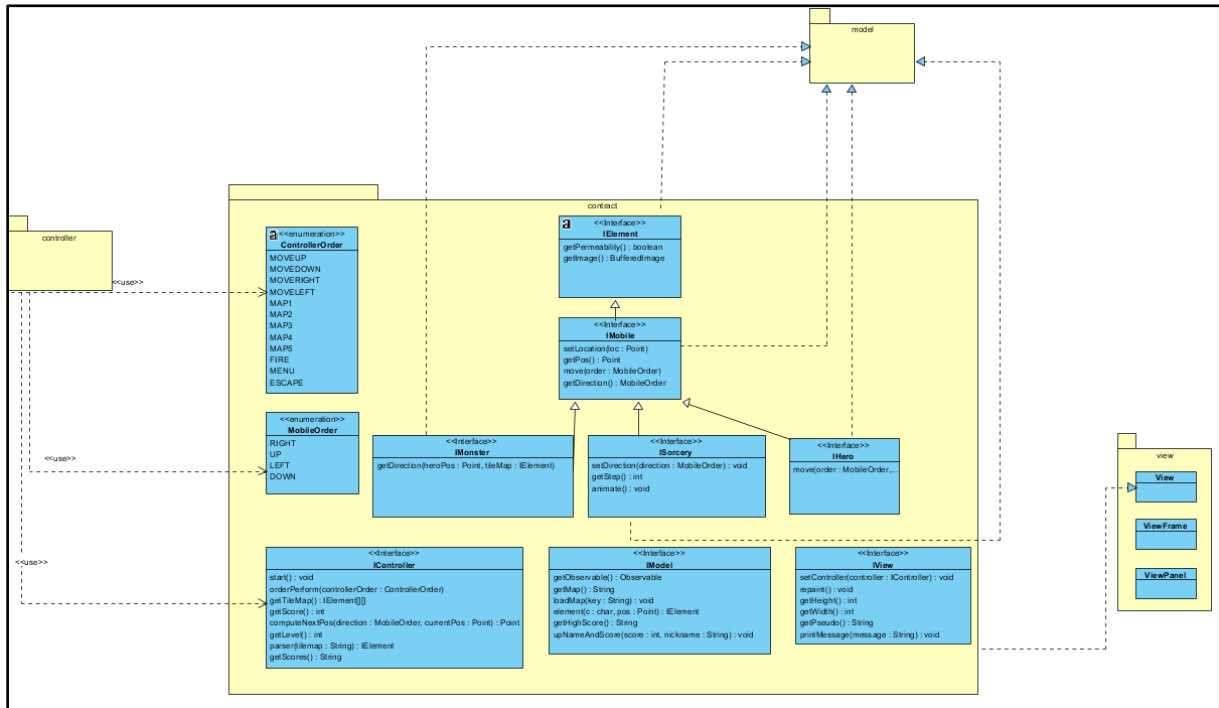
It shows the dependencies between the packages and their elements.



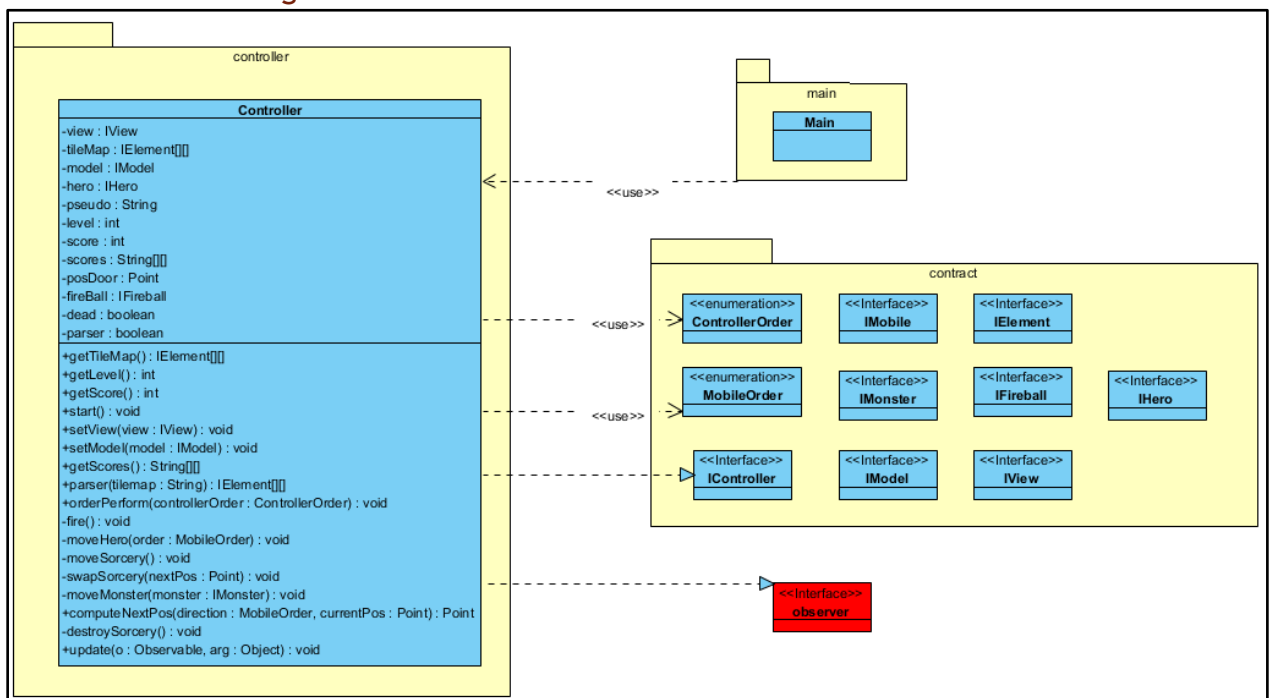
#### Class diagram

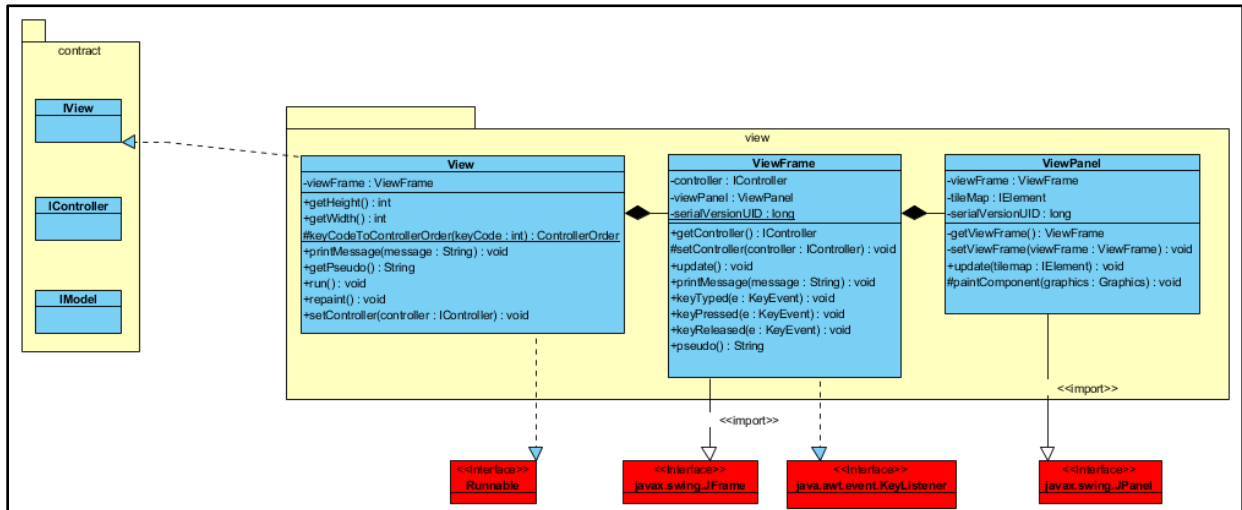
It describes the structure of a system by showing the system's classes, their attributes, operations, methods and the relationships among objects.

## Contract class diagram

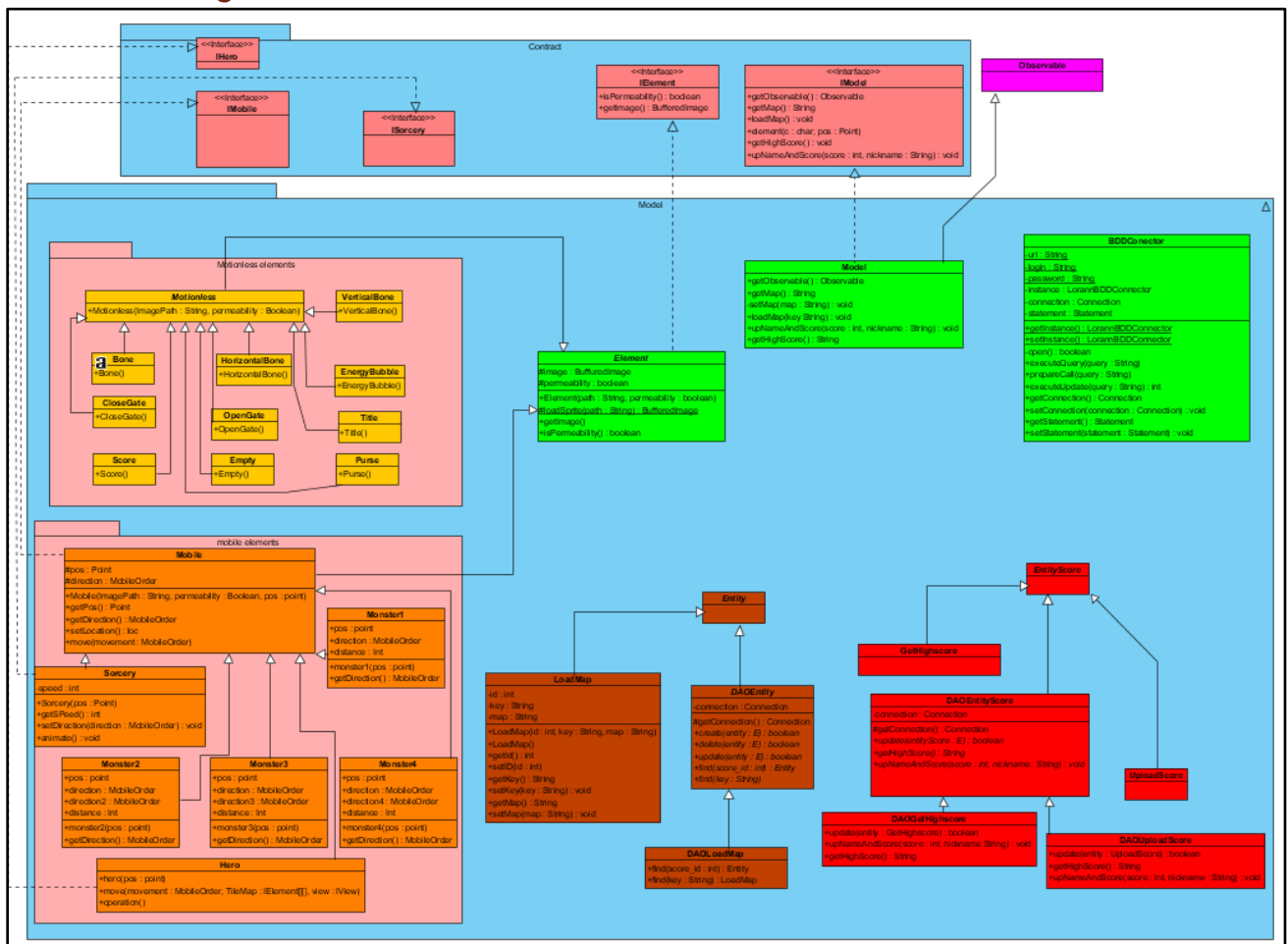


## Controller class diagram

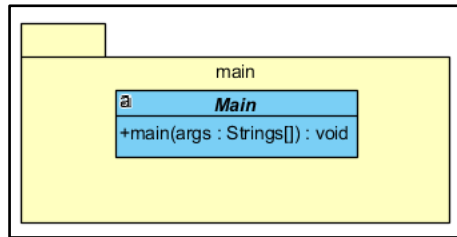


[View class diagram](#)

## Model class diagram

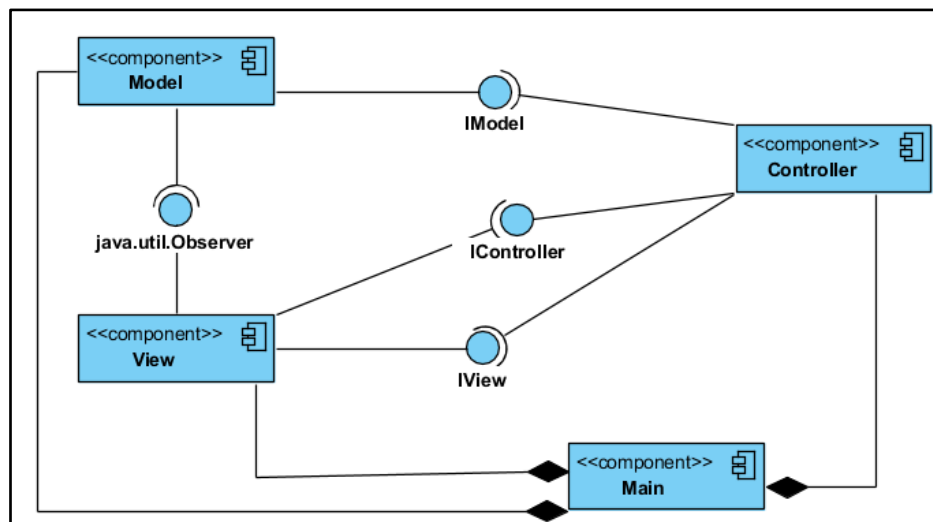


## Main class diagram



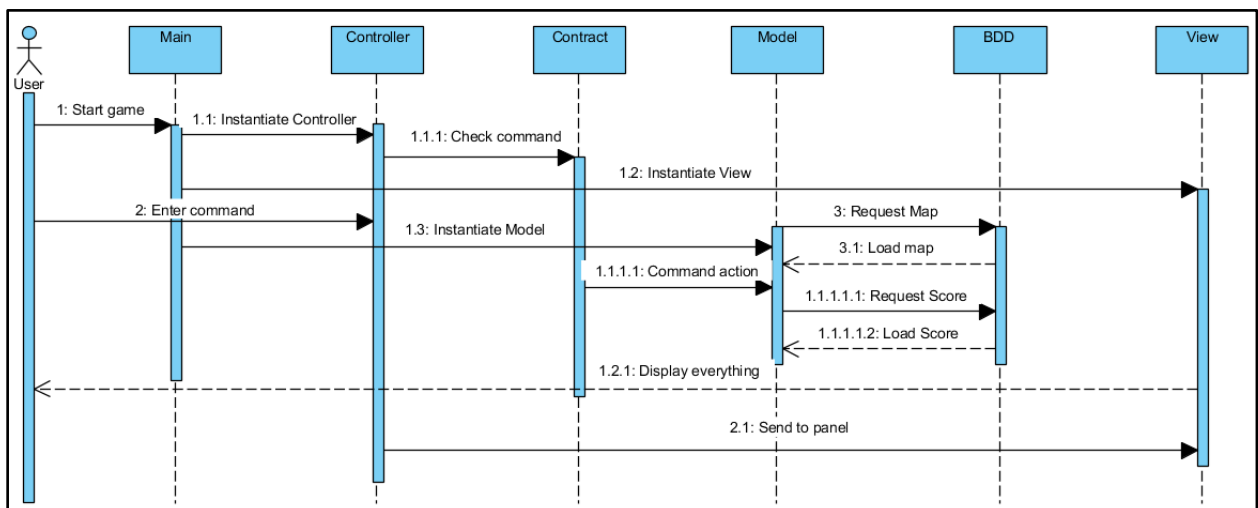
## Components diagram

It depicts how components are wired together to form larger components or software systems.



## Sequence diagram

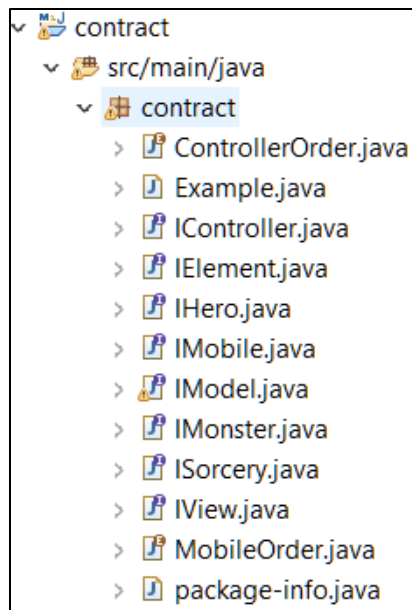
It shows object interactions in terms of exchange of message, arranged in time sequence.



## Code

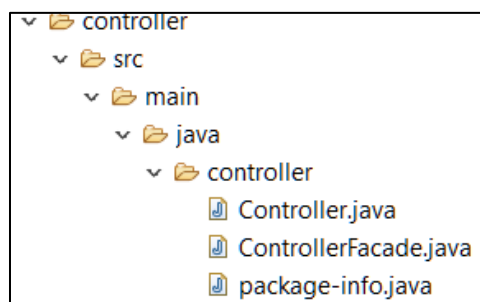
## Contract

It contains the interfaces used by the MVC.



## Controller

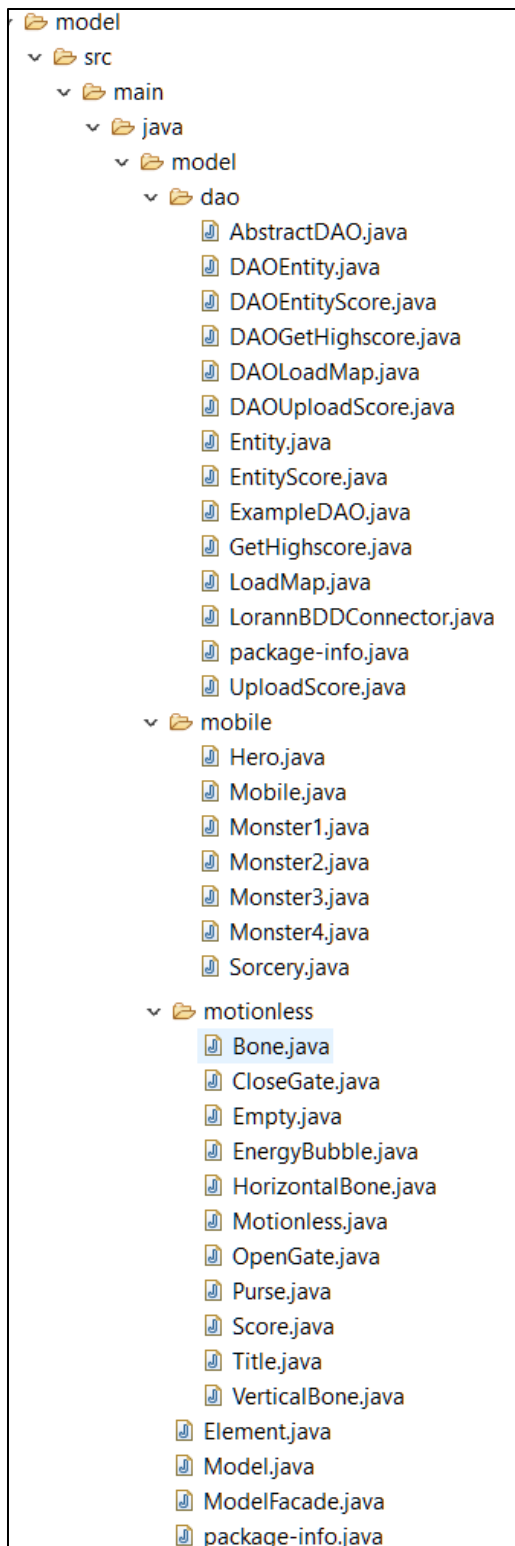
It accepts input and converts it to commands for the model or the view.



## Model

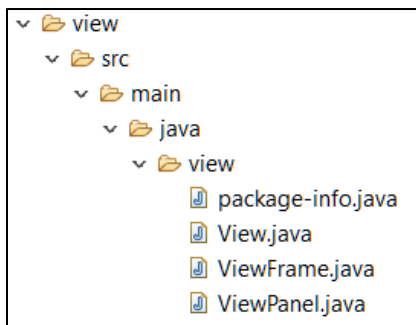
It contains the game's data.





## View

It displays a graphical interface for users, using the model's data.



## Main

It create a view, a model and a controller and it launches the game Lorann.

```

public static void main(final String[] args) {
    final Model model = new Model();
    final View view = new View(model);
    final Controller controller = new Controller(view, model);
    view.setController(controller);
    controller.start();
}
  
```

## Documentation

### Javadoc

Documentation furnishing information about the author of the source code, the code version, the attributes, the methods and their parameters, the return values...

### JXR report

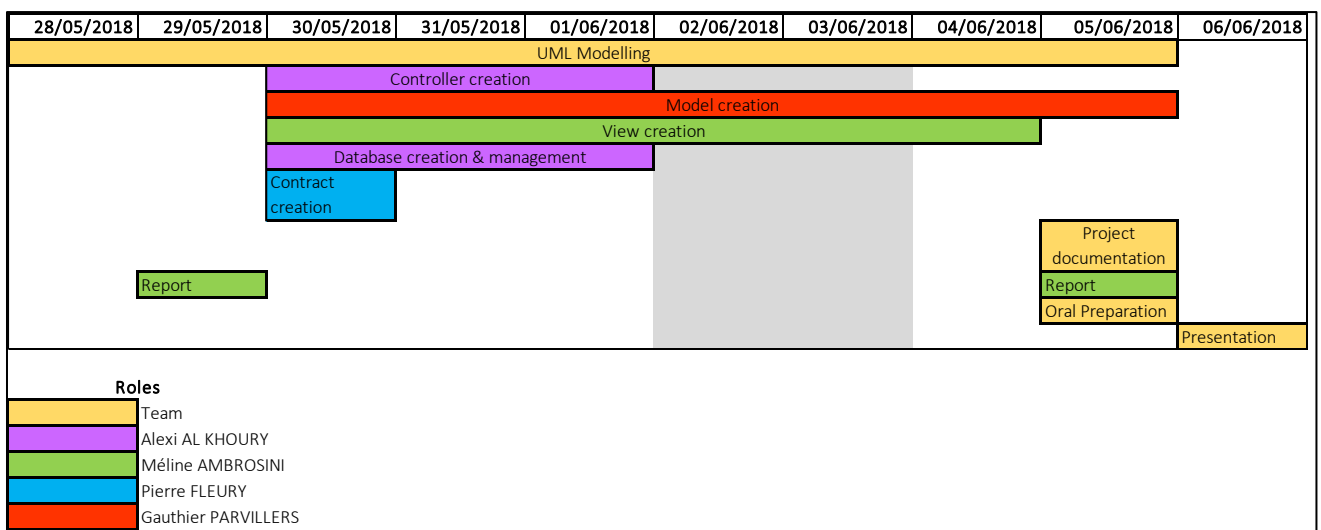
Reports that allow XHTML-based line-numbered views of Java source code.

### SureFire report

Documentation furnishing unit tests results.

## Results

### Effective schedule



## Team review

The project has progressed well but we ran out of time at the end of the project. Moreover we didn't communicate enough with each other and our schedule was a little bit messy. However, we successfully execute the game and it works properly.

## Personal review

### Méline Ambrosini

I was the project manager and in charge of the view package. I found this project more difficult than the others we've made because there was a lot to do and we didn't have a long time. Moreover, I had some flaws in Java Language so I had difficulties realizing the view.

### Alexi Al Khoury

This project was very interesting and very difficult because we did not have many prosits that work with Java, Maven and JUnit. I worked on the Database at first and then I did the controller. It made me understand how java really works.

### Pierre Fleury

This project was quite difficult for me. And I have some difficulties with the Java language. But the project was a good way to assimilate the java. Including the MVC design pattern, the syntax of the java. My team was very helpfull, and the requests were fair shared.

### Gauthier Parvillers

This project was, like the others, quite amusing in its presentation. It did make me work more on java, wich is a langage I don't have masterised. The project was quite hard, the size of the code, and the additional demands, besides the code there was the Junit and the diagrams who were very complex. But I think we could have managed it better in a lot of aspect. I just deplore the lack of communication, wich was just above the minimal.

## Ways of improvement

Among the possible improvements, we could have made a more logical planning, like for example, do the unit tests before the code. Moreover, we should have been more communicative because sometimes, we didn't know who was doing what.

For Lorann, we could have implement a way to save the game or to pause it.